

Docket No.: 67161-118

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	Customer Number: 20277
	:	
Isao MINEMATSU, et al.	:	Confirmation Number:
	:	
Serial No.:	:	Group Art Unit:
	:	
Filed: October 17, 2003	:	Examiner: Unknown
	:	
For:		SEMICONDUCTOR MEMORY DEVICE STORING PART OF PROGRAM DESIGNATED BY PROGRAMMER, AND SOFTWARE DEVELOPMENT APPARATUS FOR SYSTEM USING THE SAME

**CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT**

Mail Stop CPD
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicants hereby claim the priority of:

Japanese Patent Application No. 2003-002233, filed January 8, 2003

cited in the Declaration of the present application. A certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY



Stephen A. Becker
Registration No. 26,527

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 SAB:tlb
Facsimile: (202) 756-8087
Date: October 17, 2003

67161-118
MINEMATSU et al.
October 17, 2003

日 本 国 特 許 庁

JAPAN PATENT OFFICE

McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2003年 1月 8日

出 願 番 号

Application Number:

特願2003-002233

[ST.10/C]:

[JP2003-002233]

出 願 人

Applicant(s):

三菱電機株式会社

2003年 2月 7日

特 許 庁 長 官
Commissioner,
Japan Patent Office

太田信一郎

出証番号 出証特2003-3005649

【書類名】 特許願

【整理番号】 541745JP01

【提出日】 平成15年 1月 8日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 12/08

【発明者】

【住所又は居所】 東京都千代田区丸の内二丁目2番3号 三菱電機株式会社
社内

【氏名】 峯松 勲

【発明者】

【住所又は居所】 東京都千代田区丸の内二丁目2番3号 三菱電機株式会社
社内

【氏名】 佐藤 尚和

【特許出願人】

【識別番号】 000006013

【氏名又は名称】 三菱電機株式会社

【代理人】

【識別番号】 100064746

【弁理士】

【氏名又は名称】 深見 久郎

【選任した代理人】

【識別番号】 100085132

【弁理士】

【氏名又は名称】 森田 俊雄

【選任した代理人】

【識別番号】 100083703

【弁理士】

【氏名又は名称】 仲村 義平

【選任した代理人】

【識別番号】 100096781

【弁理士】

【氏名又は名称】 堀井 豊

【選任した代理人】

【識別番号】 100098316

【弁理士】

【氏名又は名称】 野田 久登

【選任した代理人】

【識別番号】 100109162

【弁理士】

【氏名又は名称】 酒井 將行

【手数料の表示】

【予納台帳番号】 008693

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 半導体記憶装置およびソフトウェア開発装置

【特許請求の範囲】

【請求項 1】 プロセッサと命令キャッシュとの間に接続される半導体記憶装置であって、

プロセッサによって実行される命令列の一部が格納される命令バッファと、

前記命令バッファに格納された命令列のアドレス範囲が設定されるアドレステーブルと、

前記プロセッサから出力された命令アドレスが前記アドレステーブルに設定されたアドレス範囲内であるか否かを判定する判定器と、

前記判定器の判定結果に応じて、前記命令バッファに格納される命令コードと前記命令キャッシュに格納される命令コードとを選択的に出力するセクタとを含む、半導体記憶装置。

【請求項 2】 前記判定器は、非動作モードが設定されている場合には、前記プロセッサと前記命令キャッシュとが直接接続されるように制御を行なう、請求項 1 記載の半導体記憶装置。

【請求項 3】 非動作モードが設定されている場合には、前記アドレステーブルおよび前記命令バッファが前記プロセッサのメモリマップ上にマッピングされ、メモリマップドデバイスとして動作する、請求項 1 記載の半導体記憶装置。

【請求項 4】 前記判定器は、前記プロセッサが出力した命令アドレスから、前記アドレステーブルに設定されたアドレスの最大値を減算する第 1 の減算器と、

前記プロセッサが出力した命令アドレスから、前記アドレステーブルに設定されたアドレスの最小値を減算する第 2 の減算器と、

前記第 1 の減算器および前記第 2 の減算器による減算結果の符号に応じて、前記プロセッサから出力された命令アドレスが前記アドレステーブルに設定されたアドレス範囲内であるか否かを判定する論理回路とを含む、請求項 1 記載の半導体記憶装置。

【請求項 5】 前記第 2 の減算器は、減算結果をアドレスとして前記命令バ

ッファへ出力する、請求項 4 記載の半導体記憶装置。

【請求項 6】 前記判定器はさらに、前記プロセッサから出力された命令アドレスの上位ビットと、前記アドレステーブルに設定された最大値の上位ビットとを比較し、一致しない場合には前記第 1 の減算器および前記第 2 の減算器の動作を停止させる比較器を含む、請求項 4 記載の半導体記憶装置。

【請求項 7】 前記論理回路は、前記プロセッサから出力された命令アドレスが前記アドレステーブルに設定されたアドレス範囲内であると判定した場合には、前記プロセッサから出力された命令アドレスが前記アドレステーブルに設定されたアドレス範囲外であると判定するまで前記第 2 の減算器の動作を停止させる、請求項 4 記載の半導体記憶装置。

【請求項 8】 前記プロセッサから出力される命令アドレスが前記命令バッファのアドレスとして供給される、請求項 7 記載の半導体記憶装置。

【請求項 9】 プロセッサと命令キャッシュとの間に接続され、前記プロセッサによって実行される命令列の一部を格納する半導体記憶装置を含んだプロセッサシステムのソフトウェア開発装置であって、

ソースファイルの中から、前記半導体記憶装置に配置する命令列を抽出するための抽出手段と、

前記抽出手段によって抽出された命令列が本来マップされるアドレス情報を取得するための取得手段と、

前記取得手段によって取得されたアドレス情報を前記ソースファイルと互換性のある形式に整形するための整形手段と、

前記ソースファイル、前記抽出手段によって抽出された命令列および前記整形手段によって整形されたアドレス情報からロードモジュールを生成するための生成手段とを含む、ソフトウェア開発装置。

【請求項 10】 前記抽出手段は、前記ソースファイルの中から所定の予約語で指定された命令列を抽出して、前記半導体記憶装置に配置する命令列とする、請求項 9 記載のソフトウェア開発装置。

【請求項 11】 前記抽出手段は、前記ソースファイルの中から共通命令列を抽出して、前記半導体記憶装置に配置する命令列とする、請求項 9 記載のソフ

トウェア開発装置。

【請求項 1 2】 前記取得手段は、取得したアドレス情報から命令列の先頭アドレスの下位ビットを抽出し、当該先頭アドレスの下位ビットに相当するバイト数だけ命令列のアドレスをずらす、請求項 9 ～ 1 1 のいずれかに記載のソフトウェア開発装置。

【請求項 1 3】 前記生成手段は、前記抽出手段によって抽出された命令列に対応する主記憶の領域の先頭アドレスに、前記主記憶に予め設けられた予約領域の先頭へ分岐する分岐命令を設定し、

前記抽出手段によって抽出された命令列の最後尾に、当該命令列に対応する主記憶の領域の最後尾アドレスの直後の命令へ分岐する分岐命令を追加する、請求項 9 ～ 1 1 のいずれかに記載のソフトウェア開発装置。

【発明の詳細な説明】

【 0 0 0 1 】

【発明の属する技術分野】

本発明は、CPU (Central Processing Unit) と共に使用される半導体記憶装置に関し、特に、プログラマによって指定されたプログラムの一部を格納し、CPU の命令フェッチ時に命令コードを出力する半導体記憶装置およびそれを用いたシステムのソフトウェア開発装置に関する。

【 0 0 0 2 】

【従来の技術】

近年、CPU の高速化が進み、キャッシュメモリを搭載した CPU が盛んに開発されている。一般に、キャッシュメモリは、CPU と主記憶との間に接続され、CPU が最近利用した主記憶の内容を一時的に保持する小規模、高速の記憶装置である。キャッシュは、保持する内容が CPU の命令である場合には“命令キャッシュ”、データである場合には“データキャッシュ”、命令およびデータの両方である場合には“ユニファイドキャッシュ”というように、3 種類に分類される。

【 0 0 0 3 】

キャッシュは、CPU によるメモリへのアクセスが局所的であること活用した

仕組みであり、概してCPUの処理性能を向上させる効果が得られることが広く知られている。ただし、その効果はCPUが実行するプログラムに依存する。たとえば、CPUが実行するプログラムの処理の特徴として、メモリアクセスに規則性がない場合には、キャッシュのヒット率が低くなって性能の向上を図ることができない。

【0004】

また、キャッシュは、CPUから出力された参照アドレスとキャッシュの内容との一致検出を行なう仕組みを本質的に持っているため、消費電力が大きくなる傾向にある。そのため、キャッシュを導入しても、電力効率（処理性能／消費電力）の面で必ずしも向上するとは限らない。

【0005】

さらには、キャッシュの内容はCPUによるプログラムの実行経歴に依存するため、同じアドレスにアクセスする場合でも、キャッシュにヒットする場合とミスヒットする場合とがあり、命令やデータのアクセスサイクルが保証されない。このため、キャッシュを用いたシステムにおいて、リアルタイム性の高いソフトウェアの最適化が困難である。

【0006】

これらの問題点を解決するための技術として、以下のようなものが開示されている。

【0007】

【特許文献1】

特開平10-340226号公報

【0008】

【特許文献2】

特開平9-319657号公報

【0009】

【特許文献3】

USP5,381,533

【0010】

【非特許文献 1】

P.R.Panda et al., "Efficient Utilization of Scratch-Pad Memory in Embedded Processor Applications", European Design and Test Conference, March 1997.

【0 0 1 1】

【発明が解決しようとする課題】

特許文献 1 は、タグメモリが、アドレスタグ中の各ウェイを通じて共通なビット群を有する第 1 のタグメモリと、各ウェイ毎に独自のビット群を有する第 2 のタグメモリとに分割され、データ処理装置から示されるアドレスを分割して第 1 および第 2 のタグメモリ毎に比較することにより、ヒット率を低下させることなくマイクロプロセッサの消費電力を削減するものである。しかし、電力効率の向上に寄与する反面、プログラムの実行時における性能向上には寄与しないといった問題点があった。

【0 0 1 2】

また、特許文献 2 においては、命令キャッシュが 1 ライン 1 命令で構成され、命令キャッシュと主記憶との間に命令ストリームバッファが設けられる。命令ストリームバッファは、キャッシュラインの整数倍の大きさで、主記憶から読込んだ可変長の命令列の書込みが可能であり、命令キャッシュへの出力はキャッシュライン単位で可能なように構成するものである。これによって、無駄な命令の読込みをなくすことができるが、電力効率の向上を図ることができず、命令のアクセスサイクルが保証されないといった問題点があった。

【0 0 1 3】

また、特許文献 3 においては、それぞれの命令トレースセグメントが命令のブロックを含み、それぞれのブロックの最初の命令が分岐命令の次の命令となり、それぞれのブロックの最後の命令が分岐命令となるように配置される。これによって、ヒット率を高め、電力効率を向上させるものである。しかし、特許文献 2 と同様に、命令やデータのアクセスサイクルが保証されない。

【0 0 1 4】

さらには、非特許文献 1 においては、主記憶と異なるアドレス空間に、小規模

、高速のスクラッチパッドメモリ（以下、SPMと略す。）が配置される。そして、プログラマが、頻繁にアクセスされる命令やデータを指定して、SPMに配置するものである。これによって、SPMに対するアクセスサイクル数が保証され、キャッシュに含まれる一致検出機能が不要となるため消費電力を少なくすることができる。しかし、一般に、SPMはデータメモリ用、すなわちデータキャッシュの代用とされ、命令メモリ用として実装することが困難であるといった問題点があった。

【0015】

本発明は、上記問題点を解決するためになされたものであり、その目的は、命令フェッチ時におけるアクセスサイクルを保証でき、プロセッサシステムの電力効率を向上させることが可能な半導体記憶装置およびその半導体記憶装置を含んだプロセッサシステムのソフトウェア開発を効率的に行なえるソフトウェア開発装置を提供することである。

【0016】

【課題を解決するための手段】

本発明のある局面に従えば、プロセッサと命令キャッシュとの間に接続される半導体記憶装置であって、プロセッサによって実行される命令列の一部が格納される命令バッファと、命令バッファに格納された命令列のアドレス範囲が設定されるアドレステーブルと、プロセッサから出力された命令アドレスがアドレステーブルに設定されたアドレス範囲内であるか否かを判定する判定器と、判定器の判定結果に応じて、命令バッファに格納される命令コードと命令キャッシュに格納される命令コードとを選択的に出力するセレクタとを含む。

【0017】

本発明の別の局面に従えば、プロセッサと命令キャッシュとの間に接続され、プロセッサによって実行される命令列の一部を格納する半導体記憶装置を含んだプロセッサシステムのソフトウェア開発装置であって、ソースファイルの中から、半導体記憶装置に配置する命令列を抽出するための抽出手段と、抽出手段によって抽出された命令列が本来マップされるアドレス情報を取得するための取得手段と、取得手段によって取得されたアドレス情報をソースファイルと互換性のあ

る形式に整形するための整形手段と、ソースファイル、抽出手段によって抽出された命令列および整形手段によって整形されたアドレス情報からロードモジュールを生成するための生成手段とを含む。

【0018】

【発明の実施の形態】

(第1の実施の形態)

図1は、本発明の第1の実施の形態における半導体記憶装置が利用されるプロセッサシステムの構成例を示すブロック図である。このプロセッサシステムは、半導体記憶装置（図面においてはS T A S Hと表記する。）1と、CPUコア2と、命令キャッシュ3と、データキャッシュ4と、主記憶5と、U S A R T (Universal Synchronous and Asynchronous Receiver-Transmitter) 6とを含む。

【0019】

半導体記憶装置1は、CPUコア2と命令キャッシュ3との間に接続されると共に、システムバス7にも接続される。CPUコア2は、半導体記憶装置1から直接命令をフェッチできると共に、半導体記憶装置1を介して命令キャッシュ3から命令をフェッチすることができる。また、CPUコア2は、データキャッシュ4に対してデータの読出し、書込みが可能である。なお、CPUコア2は、システムバス7を介して半導体記憶装置1にアクセスすることも可能である。

【0020】

命令キャッシュ3およびデータキャッシュ4は、ブリッジ8を介してシステムバス7に接続される。また、システムバス7には、主記憶5およびU S A R T 6などの各種の機能ブロックが接続される。図1においては、各種の機能ブロックを代表してU S A R T 6が記載されている。なお、図1においては、全ての機能ブロックが単一のL S I (Large Scale Integrated circuit) チップによって実現されているが、一部の機能が別のL S I チップによって実現されてもよいし、半導体記憶装置1がブリッジ8に直接接続される構成であってもよい。

【0021】

図2は、本発明の第1の実施の形態における半導体記憶装置1の内部構成を示すブロック図である。この半導体記憶装置1は、アドレステーブル11と、判定

器 1 2 と、命令バッファ 1 3 と、命令バッファ 1 3 から出力される命令コードおよび命令キャッシュ 3 から出力される命令コードを選択的に出力するセクタ 1 4 と、スイッチ 1（以下、SW 1 と略す。） 1 5 およびスイッチ 2（以下、SW 2 と略す。） 1 6 と、インバータ 1 7 とを含む。

【 0 0 2 2 】

命令バッファ 1 3 は、ランダムアクセスが可能なメモリデバイスであり、CPU コア 2 が実行する命令列の一部を格納する。

【 0 0 2 3 】

アドレステーブル 1 1 は、有効ビット 1 1 1 と、アドレスの最大値 1 1 2 と、アドレスの最小値 1 1 3 とが格納される領域を含む。アドレスの最大値 1 1 2 には、命令バッファ 1 3 に格納された命令列に対応するアドレスの最大値が格納される。また、アドレスの最小値 1 1 3 には、命令バッファ 1 3 に格納された命令列に対応するアドレスの最小値が格納される。有効ビット 1 1 1 には、アドレステーブルに格納されたアドレス情報が有効であるか、無効であるかを示すビットが格納される。

【 0 0 2 4 】

判定器 1 2 は、アドレステーブル 1 1 に格納されたアドレスと、CPU コア 2 から出力されたアドレスとを参照して、命令バッファ 1 3 から出力された命令コードを選択するか、命令キャッシュ 3 から出力された命令コードを選択するかを判定する。この判定器 1 2 は、減算器 1 2 1 および 1 2 2 と、減算器 1 2 1 による減算結果が格納される差レジスタ 1（1 2 3）と、減算器 1 2 2 による減算結果が格納される差レジスタ 2（1 2 4）と、インバータ 1 2 5 と、3 入力 AND 回路 1 2 6 とを含む。

【 0 0 2 5 】

減算器 1 2 1 は、CPU コア 2 が出力したアドレスからアドレスの最大値 1 1 2 を減算し、その減算結果を差レジスタ 1（1 2 3）に格納する。また、減算器 1 2 2 は、CPU コア 2 が出力したアドレスからアドレスの最小値 1 1 3 を減算し、その減算結果を差レジスタ 2（1 2 4）に格納する。

【 0 0 2 6 】

AND回路126は、有効ビット111の値と、差レジスタ1（123）の符号ビット（MSB：Most Significant Bit）と、差レジスタ2（124）の符号ビットを反転した値との論理和を演算することにより、CPUコア2が出力したアドレスが所定の範囲に含まれているか否かを判定する。すなわち、“最小値 $113 \leq \text{命令アドレス}$ ”、かつ、“命令アドレス<最大値112”、かつ、“有効ビット=1”の場合には命令アドレスがヒットしたとして、AND回路126はハイレベル（以下、Hレベルと略す。）を出力する。

【0027】

命令アドレスがヒットしたと判定された場合には、AND回路126がHレベルを出力することにより、セクタ14に命令バッファ13から出力される命令コードを選択させ、SW2（16）をオンして差レジスタ2（124）からの値（命令バッファ13のアドレス）を命令バッファ13へ出力させると共に、命令キャッシュ3へのアドレス出力が不要であるのでSW1（15）をオフする。

【0028】

また、命令アドレスがミスヒットと判定された場合には、AND回路126がロウレベル（以下、Lレベルと略す。）を出力することにより、セクタ14に命令キャッシュ3からの命令コードを選択させ、命令バッファ13へのアドレス出力が不要であるのでSW2（16）をオフすると共に、SW1（15）をオンして命令アドレスを命令キャッシュ3へ出力させる。

【0029】

図3は、CPUコア2が半導体記憶装置1をメモリマップドデバイスとしてアクセスする場合のメモリマップを示す図である。制御レジスタ“STASH_STS”は、半導体記憶装置1の動作状況（アクティブ／非アクティブ）を示すレジスタである。また、制御レジスタ“STASH_CNT”は、半導体記憶装置1の動作状況を設定するためのレジスタである。

【0030】

図3に示すメモリマップのうち、“アドレステーブル”および“命令バッファ”として示されるアドレス空間は、図2の命令バッファ13およびアドレステーブル11の写像であり、本実施の形態においては各々4Kバイトの領域となって

いる。

【0031】

これらのアドレス空間は、半導体記憶装置1が非アクティブの状態でのみ読み／書込みが可能である。制御レジスタ“STASH_CNT”に値を設定することにより、これらのアドレス空間を非アクティブ状態にしたり、アクティブ状態にしたりすることが可能である。半導体記憶装置1が非アクティブ状態の場合、判定器12は動作せず、半導体記憶装置1がバイパスされて、CPUコア2と命令キャッシュ3とが直接接続される。

【0032】

アドレステーブルの0x4000__1000番地に最大値112が設定され、0x4000__1004番地に最小値113が設定される。半導体記憶装置1のリセット直後において、アドレステーブル11は“0”に初期化される。アドレステーブル11に“0”以外の値が設定されたときに、アドレステーブル11の有効ビット111が自動的に設定される。

【0033】

図4は、本発明の第1の実施の形態における半導体記憶装置1を用いたプロセッサシステムのソフトウェア開発装置の構成を示すブロック図である。このソフトウェア開発装置は、コンピュータ本体21、ディスプレイ装置22、FD (Flexible Disk) 24が装着されるFDドライブ23、キーボード25、マウス26、CD-ROM (Compact Disc-Read Only Memory) 28が装着されるCD-ROM装置27およびネットワーク通信装置29を含む。

【0034】

ソフトウェア開発装置を実現するプログラム（以下、ソフトウェア開発プログラムと呼ぶ。）は、FD24またはCD-ROM28等の記録媒体によって供給される。ソフトウェア開発プログラムがコンピュータ本体21によって実行されることにより、ソフトウェアの開発が行なわれる。また、ソフトウェア開発プログラムは他のコンピュータよりネットワーク通信装置29を経由し、コンピュータ本体21に供給されてもよい。

【0035】

コンピュータ本体 2 1 は、CPU 3 0、ROM (Read Only Memory) 3 1、RAM (Random Access Memory) 3 2 およびハードディスク 3 3 を含む。CPU 3 0 は、ディスプレイ装置 2 2、FD ドライブ 2 3、キーボード 2 5、マウス 2 6、CD-ROM 装置 2 7、ネットワーク通信装置 2 9、ROM 3 1、RAM 3 2 またはハードディスク 3 3 との間でデータを入出力しながら処理を行う。FD 2 4 または CD-ROM 2 8 に記録されたソフトウェア開発プログラムは、CPU 3 0 により FD ドライブ 2 3 または CD-ROM 装置 2 7 を介して一旦ハードディスク 3 3 に格納される。CPU 3 0 は、ハードディスク 3 3 から適宜ソフトウェア開発プログラムを RAM 3 2 にロードして実行することによって、ソフトウェアの開発が行なわれる。

【 0 0 3 6 】

図 5 は、本発明の第 1 の実施の形態におけるソフトウェア開発装置の機能的構成を示すブロック図である。このソフトウェア開発装置は、ソースファイル（アセンブラソース）が格納されるソースファイル格納部 1 0 1 と、半導体記憶装置 1 に配置される命令列（プログラム）を抽出する命令列抽出部 1 0 2 と、命令列抽出部 1 0 2 によって抽出された命令列が本来マップされるアドレス情報を取得するアドレス情報取得部 1 0 3 と、アドレス情報取得部 1 0 3 によって取得されたアドレス情報を、アセンブラソースと互換性のある形式で整形するアドレス情報整形部 1 0 4 と、アセンブラ 1 0 5 と、リンカ 1 0 6 とを含む。

【 0 0 3 7 】

図 6 は、本発明の第 1 の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。図 6 においては、アセンブリ言語でのプログラム開発を想定している。まず、プログラマが、キーボード 2 5、マウス 2 6 などを用いて半導体記憶装置 1 に配置すべきプログラムの箇所を指定する（S 1 1）。

【 0 0 3 8 】

図 7 (a) は、CPU コア 2 によって実行されるプログラムの一部を示す図である。半導体記憶装置 1 に配置される箇所は、関数 “_f u n c 1” の中の “_s t a s h 1 _t o p” と “_s t a s h 1 _e n d” とで囲まれた部分である

。ここで、“__stash1__top”と“__stash1__end”とは、予約されたシンボルを示している。

【0039】

次に、命令列抽出部102は、ソースファイル格納部101に格納されたアセンブラソースから、半導体記憶装置1に配置されるプログラムをスクリプトプログラムで抽出する（S12）。

【0040】

図7（b）は、命令列抽出部102によって抽出されたプログラムの一例を示す図である。“__stash1__top”と“__stash1__end”との間のプログラムが抽出されている。

【0041】

次に、アセンブラ105は、ソースファイル格納部101に格納されるオリジナルのプログラムをアセンブルする。そして、リンカ106は、アセンブラ105によってアセンブルされたオリジナルのプログラムをリンクする（S13）。このアセンブルおよびリンクは、アセンブリ言語を用いた通常のソフトウェア開発と同様であるので、詳細な説明は行なわない。ただし、後述する処理に備えて、再リンク可能なようにリンクが行なわれる。

【0042】

次に、アドレス情報取得部103は、リンカ106によってリンクされたオリジナルのプログラムから、半導体記憶装置1に配置される命令列が本来マップされるアドレス情報を取得する（S14）。具体的には、リンカ106によってリンクされたプログラムをディスアセンブルするか、マップ情報を抽出することで、予約されたシンボル“__stash1__top”および“__stash1__end”のアドレス情報を取得する。

【0043】

次に、アセンブラ105は、命令列抽出部102によって抽出された命令列をアセンブルする（S15）。このアセンブルは、通常のアセンブリ言語で記述されたプログラムのアセンブルと同様であるので、詳細な説明は行なわない。

【0044】

次に、アドレス情報整形部 1 0 4 は、アドレス情報取得部 1 0 3 によって取得されたアドレス情報を、アセンブラソースと互換性のある形式に整形する（S 1 6）。図 7（c）は、アドレス情報整形部 1 0 4 によって整形されたアドレス情報を示す図である。半導体記憶装置 1 に配置されるプログラムの先頭アドレスと最終アドレスとが 2 行目に記述されている。

【 0 0 4 5 】

次に、アセンブラ 1 0 5 は、アドレス情報整形部 1 0 4 によって整形されたアドレス情報をアセンブルする（S 1 7）。このアセンブルは、通常のアセンブリ言語で記述されたプログラムのアセンブルと同様であるので、詳細な説明は行なわない。

最後に、リンカ 1 0 6 は、ステップ S 1 3，S 1 5 および S 1 7 において生成されたオブジェクトファイルをリンクして、ロードモジュールを生成する（S 1 8）。図 7（d）は、リンカ 1 0 6 によってリンクされたモジュールのディスアセンブル結果の一部を示す図である。半導体記憶装置 1 に配置される命令列と、それに対応するアドレス情報とが、それぞれ 0 x 4 0 0 0 2 0 0 0 番地と、0 x 4 0 0 0 1 0 0 0 番地とから格納されることを示している。

【 0 0 4 6 】

図 8 は、本発明の第 1 の実施の形態におけるソフトウェア開発装置の動作を、実行されるプログラムのメモリイメージとして説明する図である。図 8（a）は、標準的な開発ツールを用いて作成されたロードモジュールに対応するメモリイメージを示している。

【 0 0 4 7 】

図 8（b）は、本実施の形態におけるソフトウェア開発装置によって作成されたロードモジュールに対応するメモリイメージを示している。命令抽出部 1 0 2 によって抽出された命令列（抽出命令列）と、初期化部とがリンクされている。抽出命令列は、初期化部に含まれるプログラムが CPU コア 2 によって実行されることにより、図 8（c）に示すように半導体記憶装置 1 に設定される。なお、初期化部をロードモジュールにリンクしているが、ロードモジュールに相当する

機能をOS (Operating System) のローダで実現することも可能である。

【0048】

以上説明したように、本実施の形態における半導体記憶装置によれば、CPUコア2が実行する命令列の一部を命令バッファ13に格納し、CPUコア2が命令バッファ13に格納された命令列をフェッチするようにしたので、実行される頻度の高い命令列を命令バッファ13に格納することによって、命令キャッシュのリプレースを減らして実行サイクル数を削減でき、処理性能を向上させることが可能となった。また、命令キャッシュのアクセス頻度を減らすことができるので、電力効率を高めることが可能となった。

【0049】

また、半導体記憶装置1は、メモリマップドデバイスとしても動作するので、標準的なCPUコアであってもリソースの読出し/書込み、動作モードの設定などを行なうことが可能である。

【0050】

また、半導体記憶装置1に対するアクセスの可能性が低い場合に、半導体記憶装置1をバイパスし、CPUコア2と命令キャッシュ3とが直接接続されるように動作モードを設定することによって、半導体記憶装置1の消費電力を抑えることが可能となる。その結果、半導体記憶装置1を導入することによって発生する消費電力の増加を低減することができる。また、半導体記憶装置1が非アクティブ状態の場合においてのみ、初期化を含めた命令バッファ3の内容の書換えを可能としたので、リソースの書換えに伴う排他制御を、新たな機構を追加しなくても実現することが可能となった。

【0051】

また、命令バッファ13または命令キャッシュ3への命令アドレスの供給において、不要な方の命令アドレスの供給を停止するようにしたので、アドレスバスの変化によるメモリの不必要な状態遷移を抑制することができ、消費電力を削減することが可能となった。

【0052】

また、本実施の形態におけるソフトウェア開発装置によれば、既存のソフトウ

ェア開発ツールに、半導体記憶装置 1 に配置される命令列を抽出する機能を追加するだけで、本実施の形態における半導体記憶装置 1 に対応したソフトウェアの開発ができるようになった。すなわち、既存のソフトウェア開発ツールの簡単な拡張のみで本実施の形態におけるソフトウェア開発装置を実現することが可能となった。

【 0 0 5 3 】

また、本実施の形態における半導体記憶装置 1 を用いたプロセッサシステムは、半導体記憶装置 1 に対するアクセスサイクルが保証されるので、ソフトウェア開発者が本実施の形態におけるソフトウェア開発装置を用いることによって、リアルタイム性の高いソフトウェアを開発することが可能になった。

【 0 0 5 4 】

さらには、本実施の形態におけるソフトウェア開発装置は、アセンブラおよびリンカを含む標準的なツールチェーンを活用し、幾つかのフィルタプログラムを追加するだけで実現できるので、ソフトウェア開発プログラムを容易に作成することが可能である。したがって、本実施の形態における半導体記憶装置 1 の機能を簡単に利用することが可能となった。

【 0 0 5 5 】

（第 2 の実施の形態）

本発明の第 2 の実施の形態におけるソフトウェア開発装置の構成は、図 4 に示す実施の形態 1 におけるソフトウェア開発装置の構成と同様である。したがって、重複する構成および機能の詳細な説明は繰返さない。

【 0 0 5 6 】

図 9 は、本発明の第 2 の実施の形態におけるソフトウェア開発装置の機能的構成を示すブロック図である。図 5 に示すソフトウェア開発装置と比較して、コンパイラ 1 0 7 およびコンパイラ 1 0 7 によって生成されたオブジェクトファイルをアセンブラソースの形式に変換して整形するオブジェクトファイル整形部 1 0 8 が追加されている点のみが異なる。したがって、重複する構成および機能の詳細な説明は繰返さない。

【 0 0 5 7 】

図 1 0 は、本発明の第 2 の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。本実施の形態におけるソフトウェア開発装置の処理手順は、図 6 に示すフローチャートのステップ S 1 1 を、図 1 0 に示すステップ S 2 1 ～ S 2 3 に置換したものである。したがって、重複する処理手順の詳細な説明は繰返さない。なお、本実施の形態においては、C 言語などの高級言語を用いたプログラム開発を想定している。

【 0 0 5 8 】

まず、プログラマが、オリジナルの高級言語のソースファイルの中から、キーボード 2 5、マウス 2 6 などを用いて半導体記憶装置 1 に配置すべきプログラムの箇所を指定する（S 2 1）。

【 0 0 5 9 】

図 1 1 は、C 言語を用いたソフトウェア開発において、半導体記憶装置 1 に配置されるプログラムの一部を示す図である。半導体記憶装置 1 に配置される箇所は、“# p r a g m a _ _ s t a s h 1 _ _ t o p”と“# p r a g m a _ _ s t a s h 1 _ _ e n d”とで囲まれた部分である。ここで、“_ _ s t a s h 1 _ _ t o p”と“_ _ s t a s h 1 _ _ e n d”とは、予約された制御文字列を示している。

【 0 0 6 0 】

次に、コンパイラ 1 0 7 は、高級言語プログラムをコンパイルしてオブジェクトファイルを生成する（S 2 2）。このコンパイルは、高級言語を用いた通常のソフトウェア開発と同様であるので、詳細な説明は行なわない。

【 0 0 6 1 】

次に、オブジェクトファイル整形部 1 0 8 は、コンパイラ 1 0 7 によって生成されたオブジェクトファイルを、ソース行情報と共にディスアセンブルする。ディスアセンブルされた結果は、コンパイラ 1 0 7 が生成したアセンブリ言語レベルのプログラムに相当する情報が含まれているので、図 7（a）に示すアセンブラソースの形式に変換することができる（S 2 3）。このようにして整形され、生成されたアセンブラソースに対して、図 6 のステップ S 1 2 以降の処理が行なわれる。

【 0 0 6 2 】

以上説明したように、本実施の形態におけるソフトウェア開発装置によれば、高級言語プログラムをコンパイルしてオブジェクトファイルを生成した後、オブジェクトファイルをディスアセンブルして整形するようにしたので、本発明の第 1 の実施の形態において説明した効果に加えて、高級言語を用いたプログラムであっても、半導体記憶装置 1 に用いたプロセッサシステムに対応したソフトウェア開発を行なうことが可能となった。

【 0 0 6 3 】

(第 3 の実施の形態)

本発明の第 3 の実施の形態における半導体記憶装置は、2 組のプログラムを半導体記憶装置内に配置することが可能なものである。なお、3 組以上のプログラムを配置する半導体記憶装置は、本実施の形態における半導体記憶装置から容易に推測可能であるため、その詳細な説明は行なわない。

【 0 0 6 4 】

図 1 2 は、本発明の第 3 の実施の形態における半導体記憶装置の概略構成を示すブロック図である。この半導体記憶装置 1 a は、アドレステーブル 1 (4 1) と、アドレステーブル 2 (4 2) と、判定器 4 3 および 4 4 と、命令バッファ 1 (4 5) と、命令バッファ 2 (4 6) と、セレクトア 4 7 とを含む。

【 0 0 6 5 】

アドレステーブル 1 (4 1) およびアドレステーブル 2 (4 2) は、図 2 に示すアドレステーブル 1 1 と同じ構成および機能を有する。また、判定器 4 3 および 4 4 は、図 2 に示す判定器 1 2 と同じ構成および機能を有する。さらには、命令バッファ 1 (4 5) および命令バッファ 2 (4 6) は、命令バッファ 1 3 と同じ構成および機能を有する。

【 0 0 6 6 】

セレクトア 4 7 は、CPU コア 2 から出力された命令アドレスが、アドレステーブル 1 (4 1) に格納されたアドレスにヒットする場合、命令バッファ 1 (4 5) から出力された命令コードを選択して、CPU コア 2 へ出力する。また、セレクトア 4 7 は、CPU コア 2 から出力された命令アドレスが、アドレステーブル 2 (4 2) に格納されたアドレスにヒットする場合、命令バッファ 2 (4 6) から

出力された命令コードを選択して、CPUコア2へ出力する。さらには、セクタ47は、CPUコア2から出力された命令アドレスが、アドレステーブル1（41）に格納されたアドレスにも、アドレステーブル2（42）に格納されたアドレスにもヒットしない場合、命令キャッシュ3から出力された命令コードを選択して、CPUコア2へ出力する。

図13は、CPUコア2が半導体記憶装置1aをメモリマップドデバイスとしてアクセスする場合のメモリマップを示す図である。図3に示すメモリマップと比較して、“アドレステーブル”および“命令バッファ”がそれぞれ2つつマッピングされており、各々4Kバイトの領域となっている点のみが異なる。

【0067】

図14は、本発明の第3の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。まず、ディスアセンブラが、本実施の形態における半導体記憶装置1aを含むプロセッサシステムが実行するソフトウェアのオブジェクトファイル（a.o）51をディスアセンブルする（S31）。

【0068】

次に、ディスアセンブラによって生成されたソースファイルの中から複数箇所で発生する共通命令列が検出される（S32）。検出された共通命令列および共通命令列に関連するアドレス情報はそれぞれアセンブルされて、オブジェクトファイル（buf.o, at.o）53および55が生成される。

【0069】

最後に、生成されたオブジェクトファイル（buf.o, at.o）53および55は、オリジナルのオブジェクトファイル（a.o）51や、半導体記憶装置1の初期化のためのルーチン（stash.o）54と共にリンクされて、ロードモジュール（a.abs）56が生成される。なお、ディスアセンブラおよびリンカは、ターゲットとなるCPUコア2のソフトウェア開発用ツールがそのまま利用可能である。

【0070】

また、共通命令列の検出においては、ターゲットとなる半導体記憶装置 1 のサイズ情報 5 2 などのハードウェア構成の情報をパラメータとして用い、繰返し発生する命令パターンの中から、半導体記憶装置 1 に格納できるサイズの共通命令列が抽出される。抽出された共通命令列は、命令バッファ 1 (4 5) および命令バッファ 2 (4 6) のそれぞれに配置される。

【0 0 7 1】

以上説明したように、本実施の形態における半導体記憶装置 1 によれば、半導体記憶装置 1 内に複数の共通命令列を配置できるので、同じ動作を行なう共通プログラム部分について第 1 の実施の形態において説明した効果が得られ、さらに半導体記憶装置 1 の汎用性を高めることが可能となった。

【0 0 7 2】

(第 4 の実施の形態)

本発明の第 4 の実施の形態における半導体記憶装置は、図 2 に示す第 1 の実施の形態における半導体記憶装置と比較して、判定器の構成および機能のみが異なる。したがって、重複する構成および機能の詳細な説明は繰返さない。なお、本実施の形態における判定器の参照符号を 1 2' として説明する。

【0 0 7 3】

図 1 5 は、本発明の第 4 の実施の形態における判定器 1 2' の構成を示すブロック図である。この判定器 1 2' は、比較器 6 1 と、減算器 6 2 および 6 3 と、減算器 6 2 による減算結果が格納される差レジスタ 1 (6 4) と、減算器 6 3 による減算結果が格納される差レジスタ 2 (6 5) と、インバータ 6 6 と、2 入力 AND 回路 6 7 とを含む。

【0 0 7 4】

比較器 6 1 は、有効ビット 1 1 1 が設定されている場合、CPU コア 2 が出力したアドレスの上位 1 9 ビットと、アドレスの最大値 1 1 2 の上位 1 9 ビットとを比較する。アドレスの上位 1 9 ビットが一致する場合、比較器 6 1 は減算器 6 2 および 6 3 をイネーブルにする。また、アドレスの上位 1 9 ビットが一致しない場合には、比較器 6 1 は減算器 6 2 および 6 3 をディスエーブルにする。

【0 0 7 5】

減算器 6 2 は、CPU コア 2 が出力したアドレスの下位 1 3 ビットからアドレスの最大値 1 1 2 の下位 1 3 ビットを減算し、その減算結果を差レジスタ 1 (6 4) に格納する。また、減算器 6 3 は、CPU コア 2 が出力したアドレスの下位 1 3 ビットからアドレスの最小値 1 1 3 の下位 1 3 ビットを減算し、その減算結果を差レジスタ 2 (6 5) に格納する。

【 0 0 7 6 】

AND 回路 6 7 は、差レジスタ 1 (6 4) の符号ビットと、差レジスタ 2 (6 5) の符号ビットを反転した値との論理和を演算することにより、CPU コア 2 が出力したアドレスの下位 1 3 ビットが所定の範囲に含まれているか否かを判定する。命令アドレスの下位 1 3 ビットがヒットした場合には、AND 回路 6 7 は H レベルを出力する。

【 0 0 7 7 】

以上説明したように、本実施の形態における半導体記憶装置 1 によれば、比較器 6 1 が CPU コア 2 が出力したアドレスの上位 1 9 ビットと、アドレスの最大値 1 1 2 の上位 1 9 ビットとを比較し、比較結果に応じて減算器 6 2 および 6 3 を制御するようにしたので、減算器 6 2 および 6 3 と、差レジスタ 1 (6 4) と、差レジスタ 2 (6 5) とのビット数を削減することができ、判定器 1 2' の回路規模を削減することが可能になると共に、半導体記憶装置 1 の消費電力を削減することが可能となった。

【 0 0 7 8 】

(第 5 の実施の形態)

本発明の第 5 の実施の形態における半導体記憶装置は、図 2 に示す第 1 の実施の形態における半導体記憶装置と比較して、判定器の構成および機能のみが異なる。したがって、重複する構成および機能の詳細な説明は繰返さない。なお、本実施の形態における判定器の参照符号を 1 2'' として説明する。

【 0 0 7 9 】

図 1 6 は、本発明の第 5 の実施の形態における判定器 1 2'' の構成を示すブロック図である。この判定器 1 2'' は、減算器 7 1 および 7 2 と、減算器 7 1 による減算結果が格納される差レジスタ 1 (7 3) と、減算器 7 2 による減算結果が

格納される差レジスタ 2 (74) と、インバータ 75 と、3 入力 AND 回路 76 とを含む。

【0080】

本実施の形態においては、命令バッファ 13 に配置されるプログラムを基本ブロックに限定、すなわち命令バッファ 13 に配置される命令列に分岐命令が含まれず、命令バッファ 13 に配置される命令列への分岐も起こらないように限定を加えるものである。

【0081】

ソフトウェア開発者に対して、このような限定を加えることにより、命令バッファ 13 に対する命令コードのフェッチは必ず命令アドレスがインクリメントされることになる。すなわち、一旦命令アドレスがヒットすれば、次にミスヒットするまで差レジスタ 2 (74) の MSB の値は変化しない。そこで、次にミスヒットが発生するまで、CPU コア 2 が出力する命令アドレスからアドレスの最小値 113 を減算する減算器 72 の動作を停止させるものである。

【0082】

AND 回路 76 は、差レジスタ 1 (73) の符号ビットと、差レジスタ 2 (74) の符号ビットを反転した値との論理和を演算することにより、CPU コア 2 が出力したアドレスが所定の範囲に含まれているか否かを判定する。命令アドレスがヒットした場合には、AND 回路 76 は H レベルを出力して減算器 72 をディセーブルにして動作を停止させる。また、命令アドレスがミスヒットした場合には、AND 回路 76 が L レベルを出力して、減算器 72 をイネーブルにして動作を開始させる。

【0083】

また、命令バッファ 13 には、CPU コア 2 が出力する命令アドレスの下位 12 ビットが供給される。

【0084】

図 17 は、本発明の第 5 の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。本実施の形態におけるソフトウェア開発装置の処理手順は、図 6 に示すフローチャートのステップ S15 を、図 17

に示すステップ S 3 1 ~ S 3 3 に置換したものである。したがって、重複する処理手順の詳細な説明は繰返さない。

【 0 0 8 5 】

アドレス情報取得部 1 0 3 は、ステップ S 1 4 において取得したアドレス情報から、先頭命令のアドレスの下位 1 2 ビットを取得する (S 3 1) 。そして、命令列抽出部 1 0 2 によって抽出された命令列 (プログラム) を、先頭命令のアドレスの下位 1 2 ビットに相当するバイト数だけ命令バッファ 1 3 内においてずらす。このようにして、 C P U コア 2 から出力される命令アドレスの下位 1 2 ビットと、命令バッファ 1 3 のアドレスの下位 1 2 ビットとの整合をとるようにしている。

【 0 0 8 6 】

次に、アセンブラ 1 0 5 は、命令列抽出部 1 0 2 によって抽出された命令列をアセンブルし (S 3 3) 、図 6 のステップ S 1 6 の処理へ進む。

【 0 0 8 7 】

以上説明したように、ソフトウェア開発者にプログラム作成時に制限を課すことによって、判定器 1 2 ” における判定を容易に行なうことができると共に、命令アドレスがヒットしている間は減算器 7 2 の動作を停止させることができるので、半導体記憶装置 1 の消費電力を削減することが可能となった。

【 0 0 8 8 】

(第 6 の実施の形態)

ソフトウェアのデバッグで広く使用されているブレークポイント機能は、通常、ブレークポイントに相当する命令を例外命令に置換することによって実現される。しかし、本発明の第 1 の実施の形態における半導体記憶装置 1 においては、プロセッサシステムが動作中に命令バッファ 1 3 などの書換えが不可能であり、命令バッファ 1 3 に例外命令を設定することができない。したがって、プログラムのデバッグが困難になるといった問題点がある。本実施の形態においては、この問題点を解消してプログラムのデバッグを効率良く行なえるようにしたものである。

【 0 0 8 9 】

本発明の第 6 の実施の形態における半導体記憶装置を用いたプロセッサシステムの構成は、第 5 の実施の形態における半導体記憶装置を用いたプロセッサシステムの構成と比較して、半導体記憶装置用予約領域がメモリマップにマッピングされている点のみが異なる。したがって、重複する構成および機能の詳細な説明は繰返さない。

【 0 0 9 0 】

図 1 8 は、本発明の第 6 の実施の形態におけるメモリマップの一例を示す図である。主記憶 5 に、半導体記憶装置 1 の命令バッファ 1 3 と同じサイズの半導体記憶装置用予約領域が設定される。

【 0 0 9 1 】

本発明の第 6 の実施の形態におけるソフトウェア開発装置の処理手順は、第 5 の実施の形態におけるソフトウェア開発装置の処理手順と比較して、ステップ S 1 8 の処理を除いて同じである。したがって、重複する処理手順の詳細な説明は繰返さない。なお、本実施の形態において、ステップ S 1 8 をステップ S 1 8' として説明する。

【 0 0 9 2 】

ステップ S 1 8' において、リンカ 1 0 6 は、命令バッファ 1 3 に配置される命令列に対応する主記憶 5 の領域の先頭アドレスに分岐命令を設定する。その分岐命令の分岐先は、図 1 8 示す半導体記憶装置用予約領域の先頭アドレスとする。また、命令列抽出部 1 0 2 によって抽出された命令列（抽出命令列）の末尾に分岐命令を追加して半導体記憶装置用予約領域に設定する。その分岐命令の分岐先は、抽出命令列に対応する主記憶領域の最後尾アドレスの直後に設定する。

【 0 0 9 3 】

この処理を、図 7 に示すプログラムを用いて具体的に説明する。まず、ステップ S 1 8' の前半の処理によって、0 x 2 6 4 番地に 0 x f f 0 0 0 への分岐命令が設定される。そして、ステップ S 1 8' の後半の処理によって、図 7 (b) に示す抽出命令列の直後に 0 x 2 6 c 番地への分岐命令が追加され、この命令列が 0 x f f 0 0 0 番地に設定される。

【 0 0 9 4 】

このようにして、デバッグ用の初期化部をリンクした場合、分岐命令以外の抽出命令列本体の実行は、主記憶 5 の半導体記憶装置用予約領域からの命令フェッチによって行なわれる。この領域は主記憶 5 上にあるので、ブレークポイントの設定が可能である。すなわち、プログラム本体と抽出命令列はそのまま、初期化部だけ異なるプログラムを利用すれば、プログラム本体と抽出命令列のアルゴリズムをそのまま用いて、ソフトウェアのデバッグが行なえるようになる。

【 0 0 9 5 】

以上説明したように、本実施の形態におけるプロセッサシステムによれば、半導体記憶装置用予約領域を主記憶 5 上に設け、抽出命令列を半導体記憶装置用予約領域に配置するようにしたので、ブレークポイントを設定することができ、ソフトウェアのデバッグを容易に行なうことが可能となった。

【 0 0 9 6 】

今回開示された実施の形態は、すべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

【 0 0 9 7 】

【発明の効果】

本発明の半導体記憶装置によれば、セレクタが、判定器の判定結果に応じて、命令バッファに格納される命令コードと命令キャッシュに格納される命令コードとを選択的に出力するので、プロセッサが命令バッファに格納された命令をフェッチする場合にはアクセスサイクルが保証されることでリアルタイム性の高いソフトウェアの開発が容易となると共に、命令キャッシュの動作が行なわれないので電力効率を向上させることが可能となった。

【 0 0 9 8 】

本発明のソフトウェア開発装置によれば、抽出手段が、ソースファイルの中から、半導体記憶装置に配置する命令列を抽出する。そして、生成手段が、ソースファイル、抽出手段によって抽出された命令列および整形手段によって整形されたアドレス情報からロードモジュールを生成するので、プログラマは半導体記憶

装置を含んだプロセッサシステムのソフトウェア開発を容易に行なうことが可能となった。

【図面の簡単な説明】

【図 1】 本発明の第 1 の実施の形態における半導体記憶装置が利用されるプロセッサシステムの構成例を示すブロック図である。

【図 2】 本発明の第 1 の実施の形態における半導体記憶装置 1 の内部構成を示すブロック図である。

【図 3】 CPU コア 2 が半導体記憶装置 1 をメモリマップドデバイスとしてアクセスする場合のメモリマップを示す図である。

【図 4】 本発明の第 1 の実施の形態における半導体記憶装置 1 を用いたプロセッサシステムのソフトウェア開発装置の構成を示すブロック図である。

【図 5】 本発明の第 1 の実施の形態におけるソフトウェア開発装置の機能的構成を示すブロック図である。

【図 6】 本発明の第 1 の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。

【図 7】 (a) は、CPU コア 2 によって実行されるプログラムの一部を示す図である。(b) は、命令列抽出部 102 によって抽出されたプログラムの一例を示す図である。(c) は、アドレス情報整形部 104 によって整形されたアドレス情報を示す図である。(d) は、リンカ 106 によってリンクされたモジュールのディスアセンブル結果の一部を示す図である。

【図 8】 本発明の第 1 の実施の形態におけるソフトウェア開発装置の動作を、実行されるプログラムのメモリエイメージとして説明する図である。

【図 9】 本発明の第 2 の実施の形態におけるソフトウェア開発装置の機能的構成を示すブロック図である。

【図 10】 本発明の第 2 の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。

【図 11】 C 言語を用いたソフトウェア開発において、半導体記憶装置 1 に配置されるプログラムの一部を示す図である。

【図 12】 本発明の第 3 の実施の形態における半導体記憶装置の概略構成

を示すブロック図である。

【図 1 3】 CPUコア 2 が半導体記憶装置 1 a をメモリマップドデバイスとしてアクセスする場合のメモリマップを示す図である。

【図 1 4】 本発明の第 3 の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。

【図 1 5】 本発明の第 4 の実施の形態における判定器 1 2' の構成を示すブロック図である。

【図 1 6】 本発明の第 5 の実施の形態における判定器 1 2'' の構成を示すブロック図である。

【図 1 7】 本発明の第 5 の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。

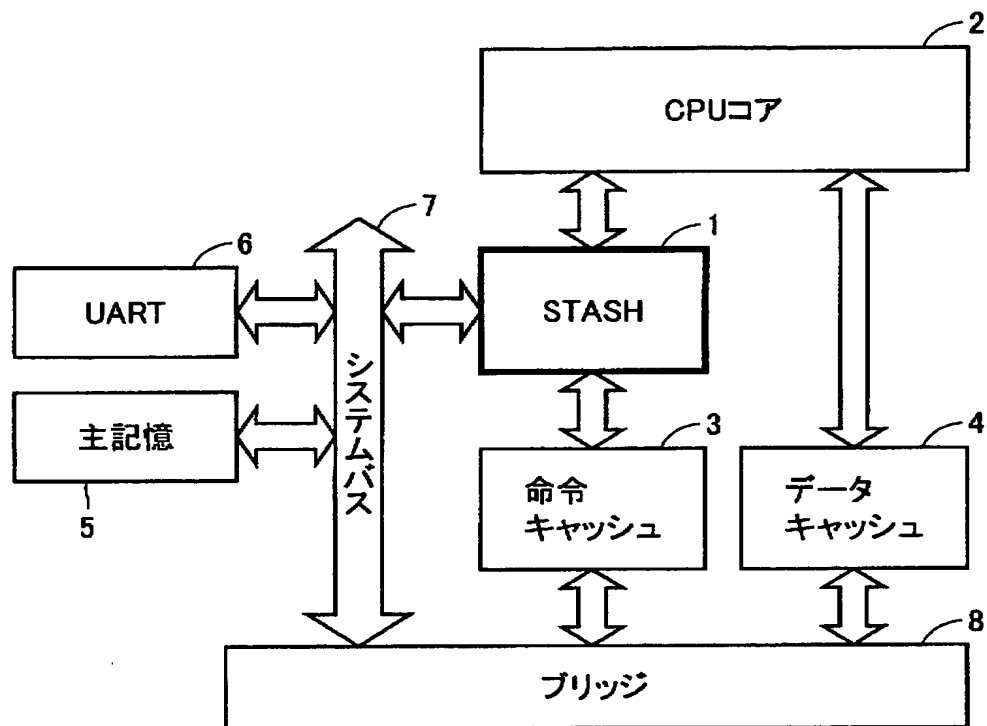
【図 1 8】 本発明の第 6 の実施の形態におけるメモリマップの一例を示す図である。

【符号の説明】

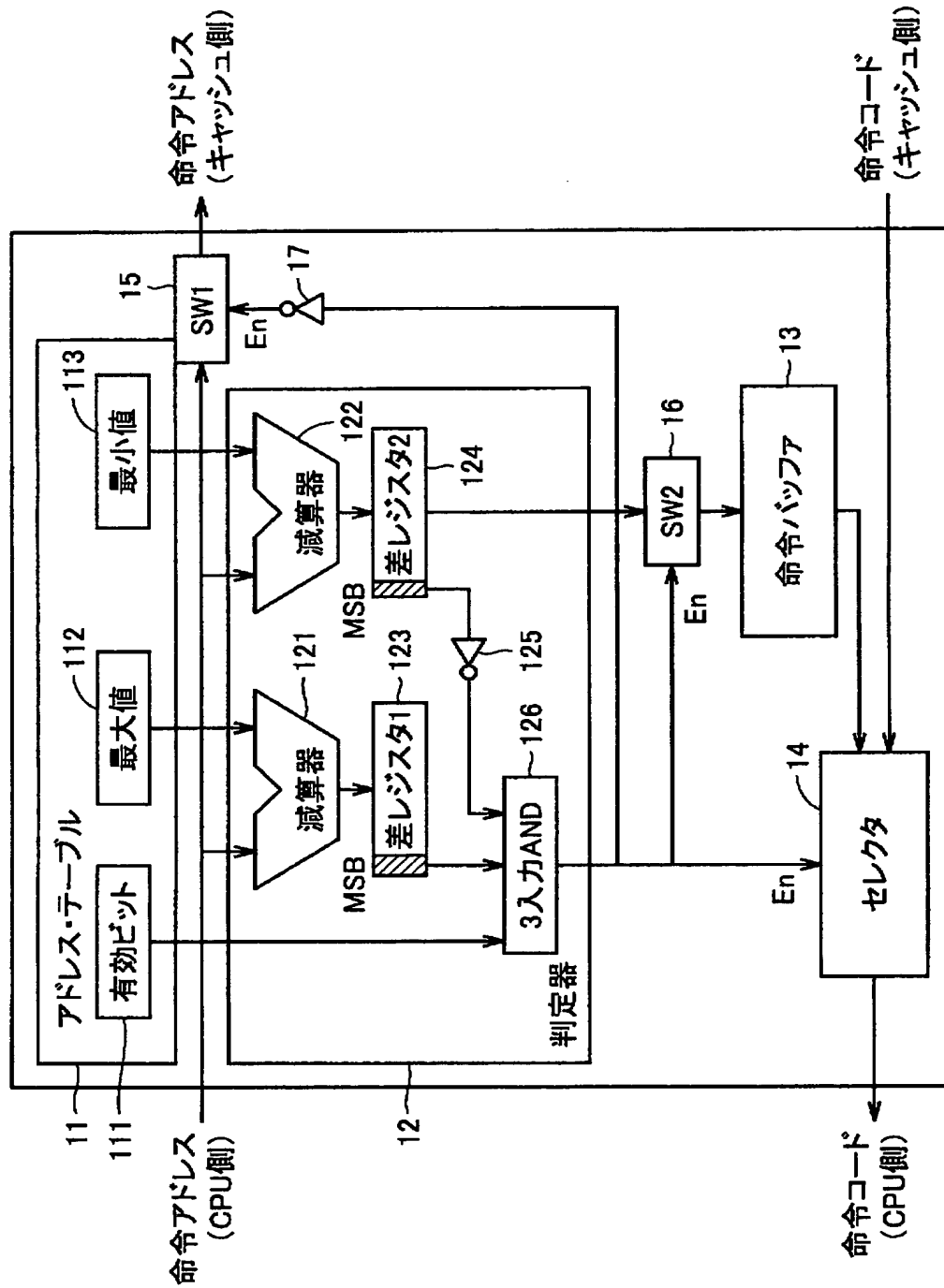
1 半導体記憶装置、2 CPUコア、3 命令キャッシュ、4 データキャッシュ、5 主記憶、6 USART、7 システムバス、8 ブリッジ、1 1, 4 1, 4 2 アドレステーブル、1 2, 1 2', 1 2'', 4 3, 4 4 判定器、1 3, 4 5, 4 6 命令バッファ、1 4, 4 7 セレクタ、1 5, 1 6 SW、1 7, 6 6, 7 5, 1 2 5 インバータ、2 1 コンピュータ本体、2 2 ディスプレイ装置、2 3 FDドライブ、2 4 FD、2 5 キーボード、2 6 マウス、2 7 CD-ROM装置、2 8 CD-ROM、2 9 ネットワーク通信装置、5 1 オブジェクト、5 2 サイズ情報、5 3 共通命令列、5 4 初期化部、5 5 アドレス情報、5 6 ロードモジュール、6 1 比較器、6 2, 6 3, 7 1, 7 2, 1 2 1, 1 2 2 減算器、6 4, 6 5, 7 3, 7 4, 1 2 3, 1 2 4 差レジスタ、6 7, 7 6, 1 2 6 AND、1 0 1 ソースファイル格納部、1 0 2 命令列抽出部、1 0 3 アドレス情報取得部、1 0 4 アドレス情報整形部、1 0 5 アセンブラ、1 0 6 リンカ、1 0 7 コンパイラ、1 0 8 オブジェクトファイル整形部、1 1 1 有効ビット、1 1 2 最大値、1 1 3 最小値。

【書類名】 図面

【図 1】



【図 2】



【図 3】

アドレス

0x4000_0000

0x4000_0004

0x4000_1000

⋮

0x4000_1FFF

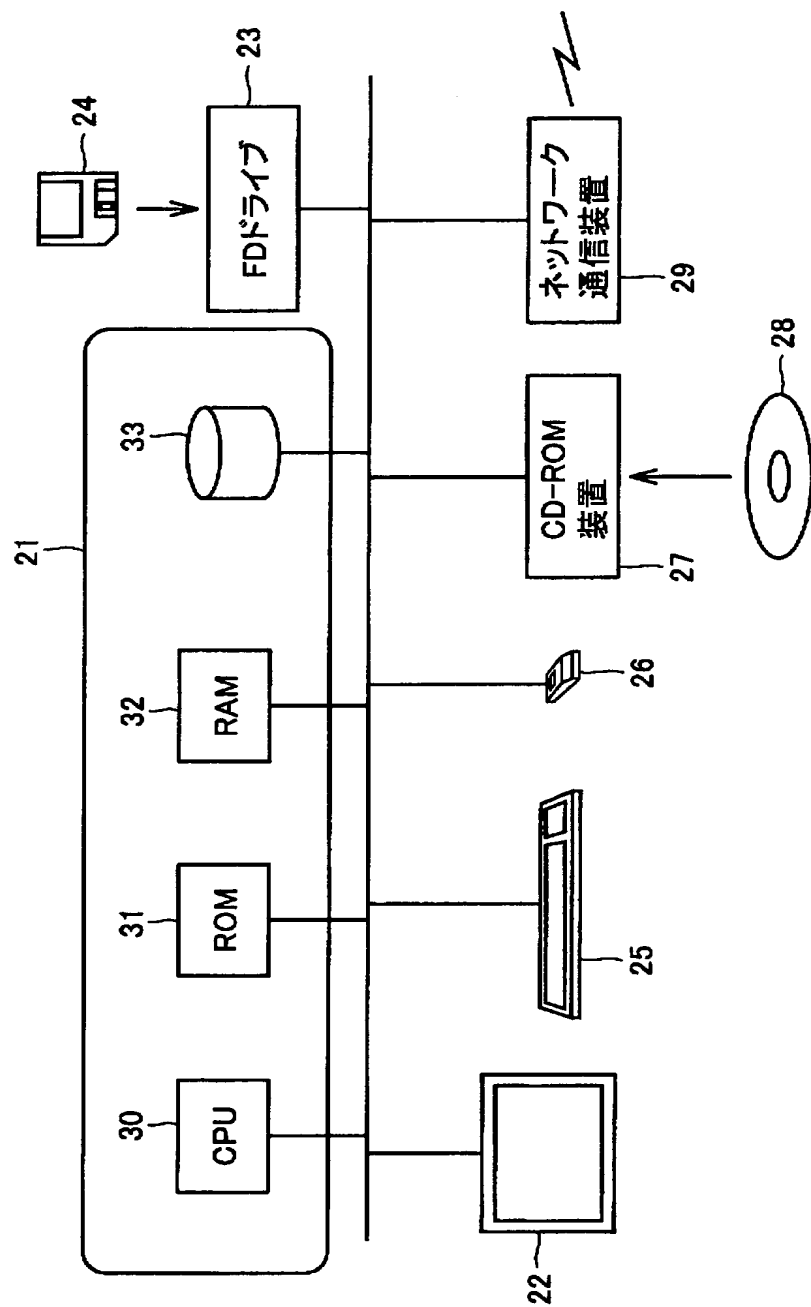
0x4000_2000

⋮

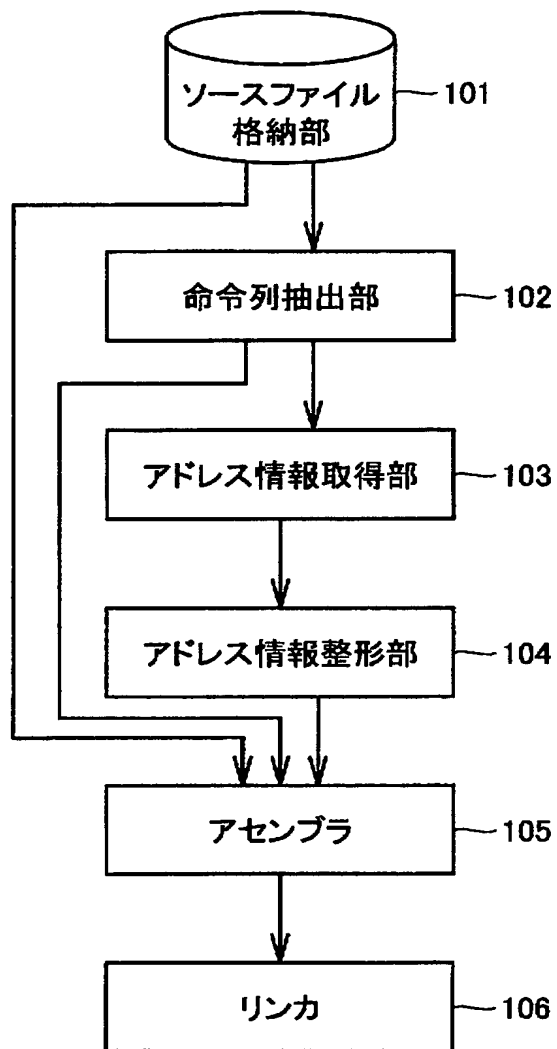
0x4000_2FFF

STASH_CNT
STASH_STS
(使用禁止)
アドレステーブル
命令バッファ

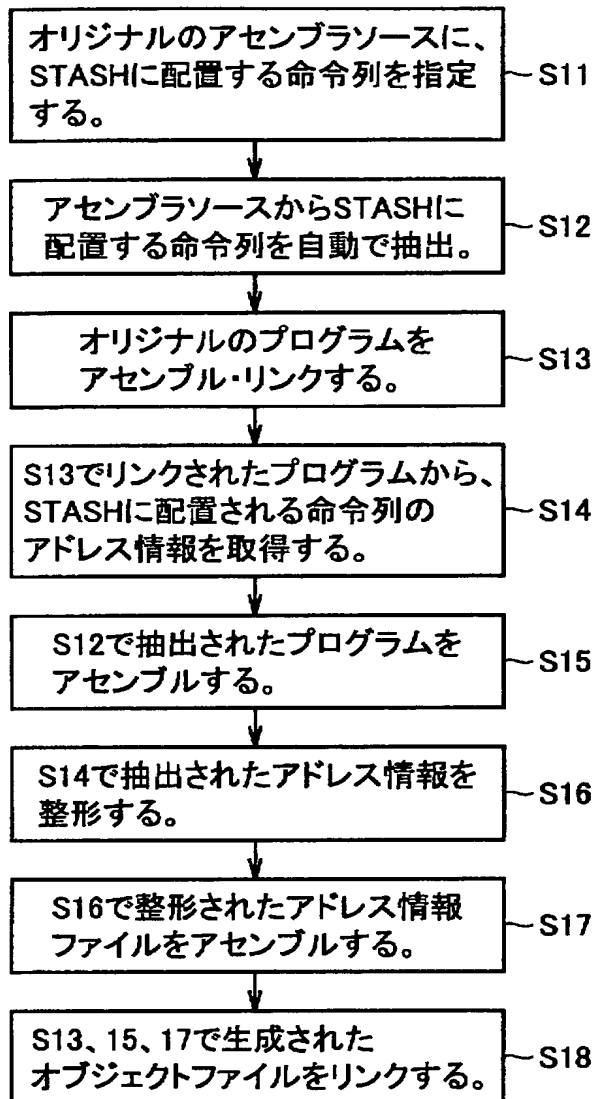
【図 4】



【図 5】



【図 6】



【図 7】

(a)

```
.text
_func1:
_stash1_top:
  ld r1, @r14+
  ld r2, @r14+
_stash1_end:
  add r1, r2
  (以下略)
```

(b)

```
ld r1, @r14+
ld r2, @r14+
```

(c)

```
.section _stash_at, "a"
.word 0x00000264, 0x0000026c
```

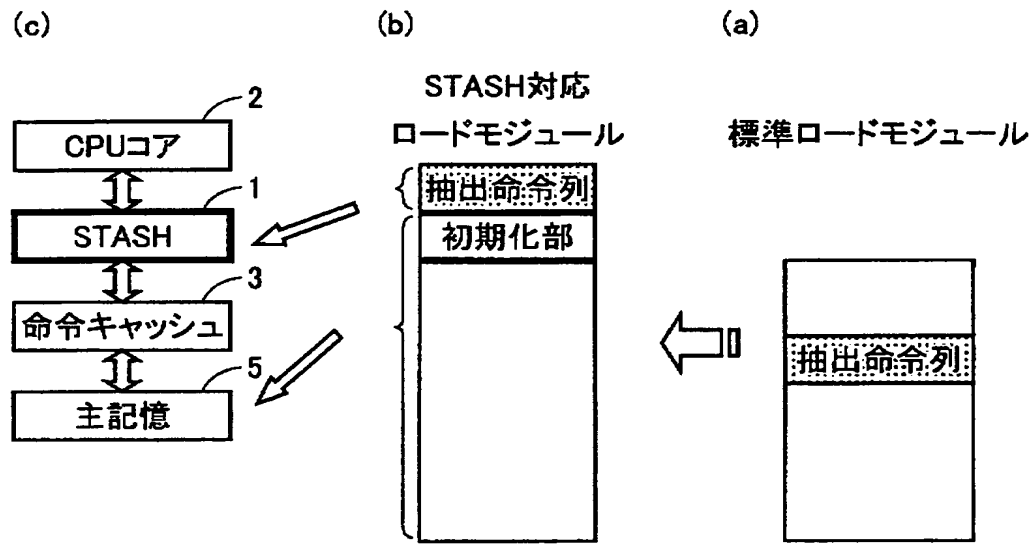
(d)

```
Disassembly of section .stash_at:

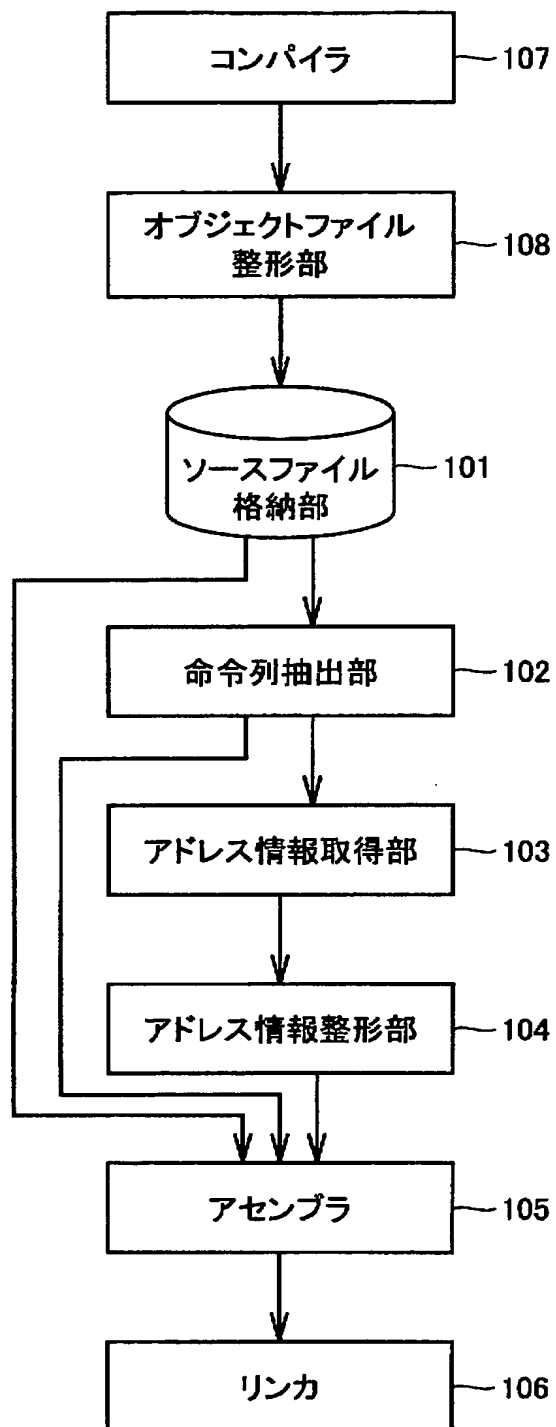
40001000 <.stash_at>:
    0: 00 00 02 64      subv r0,r0→ cmpeq r2,r4
    4: 00 00 02 6c      subv r0,r0→ cmpeq r2,r12
Disassembly of section .stash_buf:

40002000 <.stash_buf>:
    0: 21 ee 22 ee      ld r1,@lr→ ld r2,@lr+
```

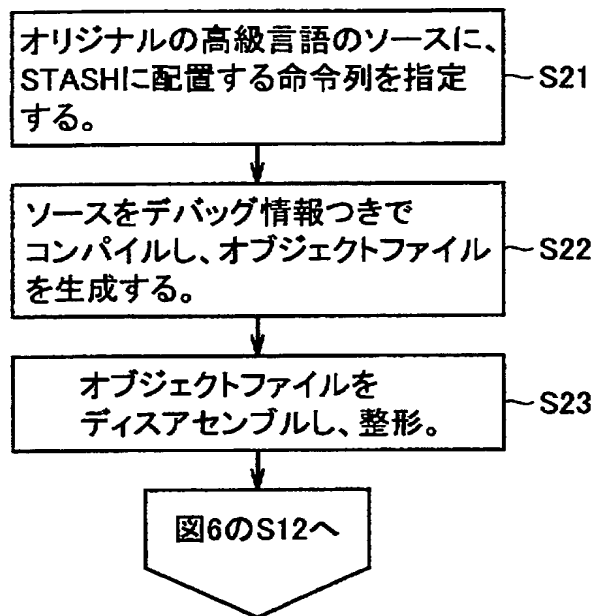
【図 8】



【図 9】



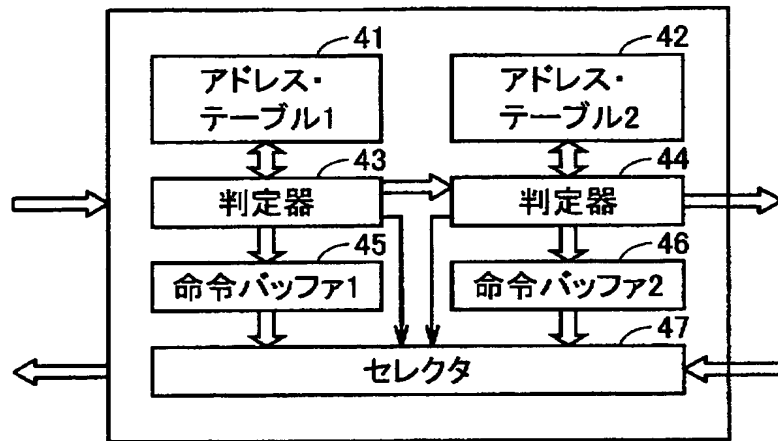
【図 1 0】



【図 1 1】

```
for(i= 0;; i++){  
#pragma _stash1_top      MACRO_A(i, j);  
    if(j > INT_MAX)  
        break;  
}  
#pragma _stash1_end  
return(j);
```

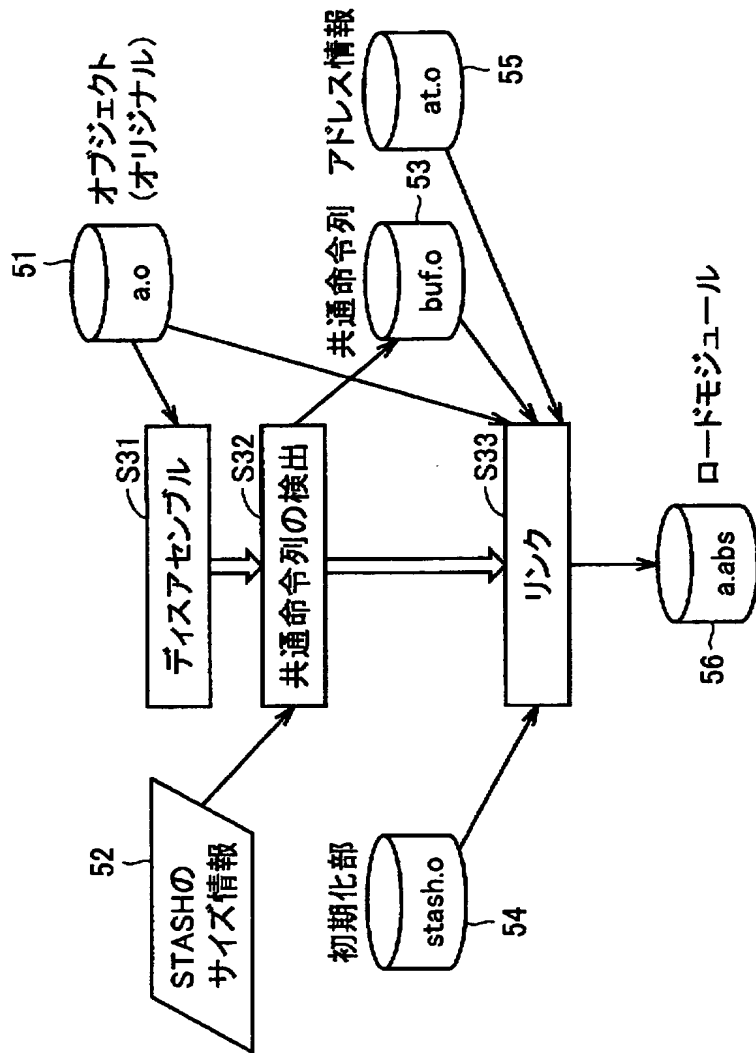

【図 1 2】



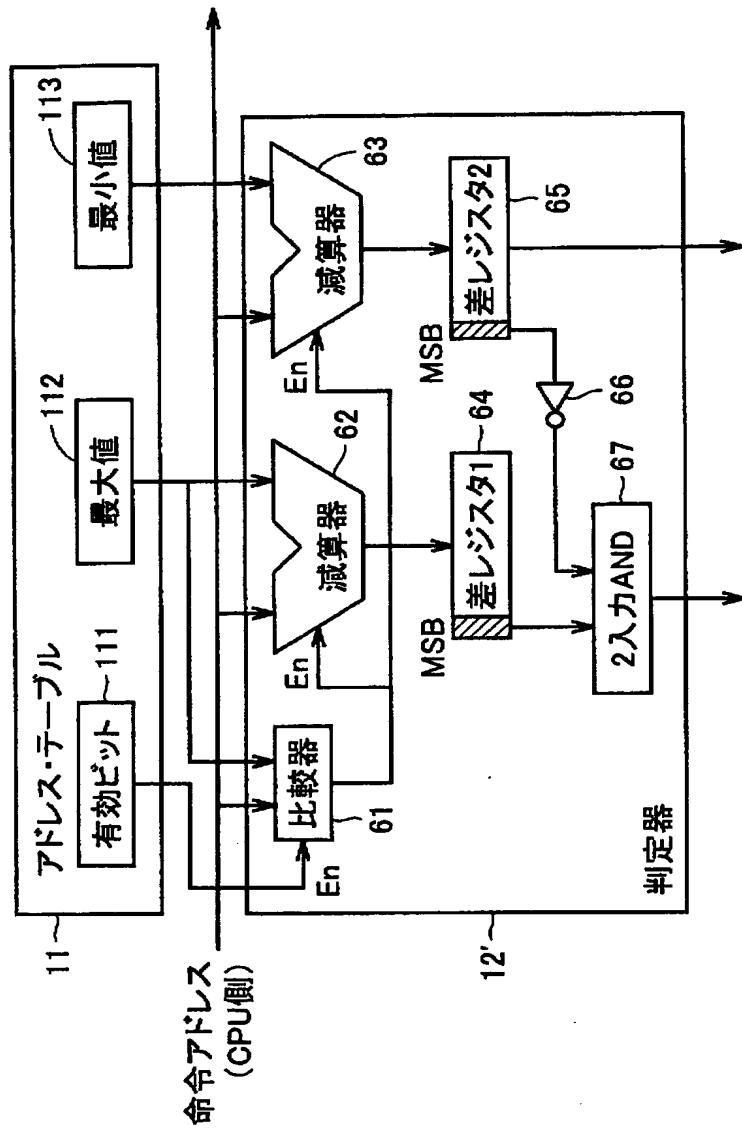
【図 1 3】

アドレス	
0x4000_0000	STASH_CNT
0x4000_0004	STASH_STS
	(使用禁止)
0x4000_1000	アドレステーブル1
}	
0x4000_1FFF	
0x4000_2000	アドレステーブル2
}	
0x4000_2FFF	
0x4000_3000	命令バッファ1
}	
0x4000_3FFF	
0x4000_4000	命令バッファ2
}	
0x4000_4FFF	

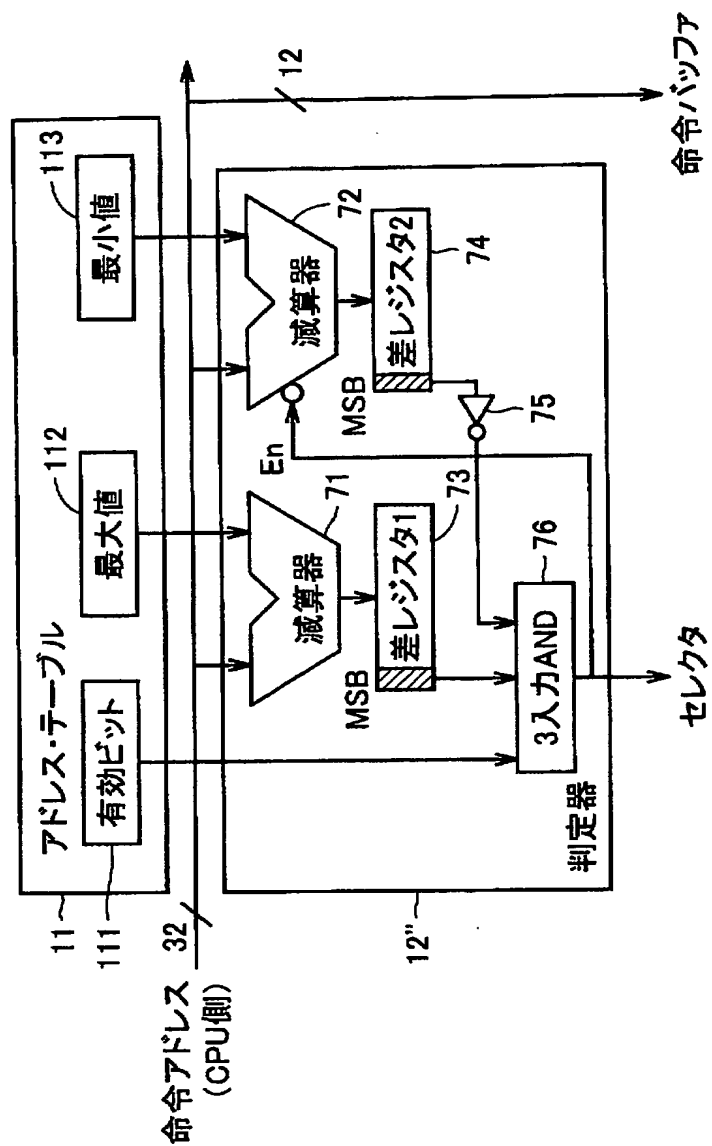
【図 14】



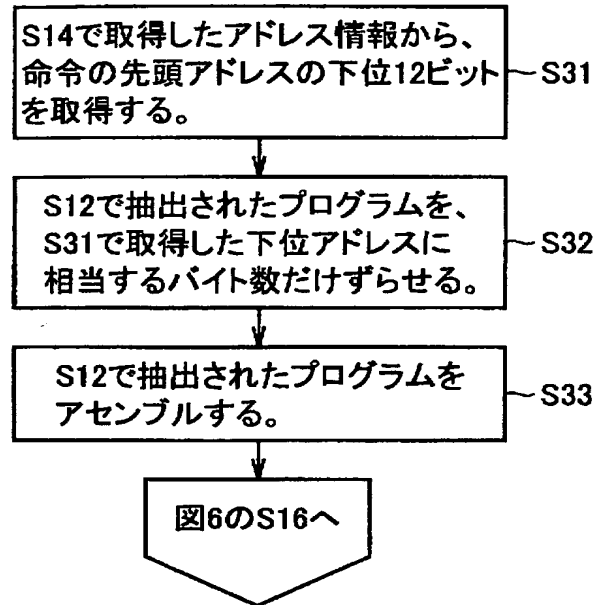
【図 1 5】



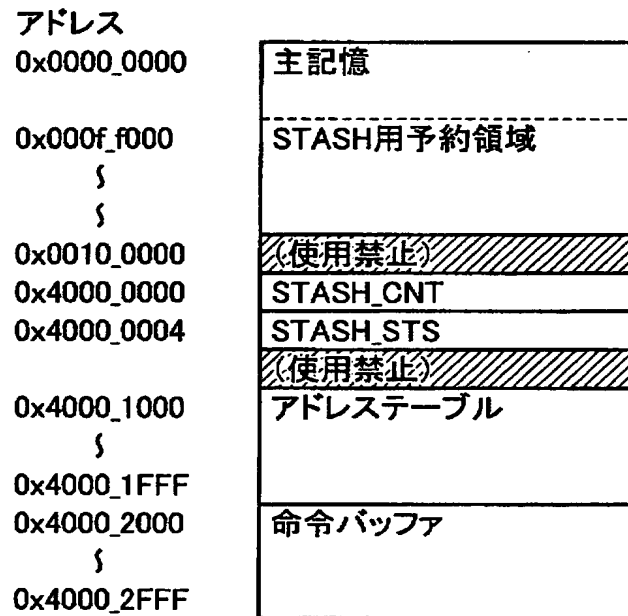
【図 16】



【図 1 7】



【図 1 8】



【書類名】 要約書

【要約】

【課題】 命令フェッチ時におけるアクセスサイクルを保証でき、プロセッサシステムの電力効率を向上させることが可能な半導体記憶装置を提供すること。

【解決手段】 アドレステーブル 1 1 に、命令バッファ 1 3 に格納された命令列のアドレス範囲が設定される。判定器 1 2 は、CPU コアから出力された命令アドレスがアドレステーブル 1 1 に設定されたアドレス範囲内であるか否かを判定する。そして、セクタ 1 4 が、判定器 1 2 の判定結果に応じて、命令バッファ 1 3 に格納される命令コードと命令キャッシュに格納される命令コードとを選択的に出力する。したがって、CPU コアが命令バッファ 1 3 に格納された命令をフェッチする場合にはアクセスサイクルが保証されると共に、命令キャッシュの動作が行なわれないので電力効率を向上させることが可能となる。

【選択図】 図 2

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 6 0 1 3]

1. 変更年月日 1 9 9 0 年 8 月 2 4 日

[変更理由] 新規登録

住 所 東京都千代田区丸の内2丁目2番3号

氏 名 三菱電機株式会社